

FACULTATEA CALCULATOARE, INFORMATICA SI MICROELECTRONICA
UNIVERSITATEA TEHNICA A MOLDOVEI

MEDII INTERACTIVE DE DEZVOLTARE A PRODUSELOR SOFT
LUCRAREA DE LABORATOR#1

Version Control Systems

Autor:

Scripnic ALEXANDRU

lector asistent:

Irina COJANU

lector superior:

Svetlana COJOCARU

Laboratory work #2

1 Scopul lucrării de laborator

Studierea unui Version Control Systems și lucrul cu versiunea CLI a acestuia. Examinarea instrucțiunilor de bază și lucru cu un repository remote.

2 Obiective

- Înțelegerea și folosirea CLI (basic level)
- Administrarea remote a mașinilor linux machine folosind SSH (remote code editing)
- Version Control Systems (git — bitbucket — mercurial — svn)

3 Laboratory work implementation

3.1 Tasks and Points

- initializeaza un nou repository
- configureaza-ti VCS
- crearea branch-urilor (creeaza cel putin 2 branches)
- commit pe ambele branch-uri (cel putin 1 commit per branch)
- seteaza un branch to track a remote origin pe care vei putea sa faci push (ex. Github, Bitbucket or custom server)
- reseteaza un branch la commit-ul anterior
- folosirea fisierului .gitignore
- merge 2 branches
- rezolvarea conflictelor a 2 branches
- folosirea tag-urilor pentru marcarea schimbarilor semnificative precum release-ul.

3.2 Analiza lucrarii de laborator

Repository link

Primul pas a fost crearea unui nou repository prin intermediul UI-ului oferit de github, iar mai precis apăsând pe butonul **New** de pe pagina utilizatorului, tab-ul cu denumirea **Repositories**. După setarea numelui pentru repository, am apăsât **create repository**.

Al doilea pas a constituit în crearea unui ssh key și l-am copiat în lista mea de key in account-ul github, ce mi-a permis editarea datelor din repositoryul creat anterior [1]. Apoi am clonat repositoryul local prin intermediul la instrucțiunea: `git clone git@github.com:AScripnic/MIDPS-laboratories.git`.

Mi-am configurat config-ul la git cli prin intermediul la instrucțiunile `git config user.name "ascripnic"` și `git config user.email "scripnic.alexandru@gmail.com"` [2].

În al treilea pas am create 2 branchuri și câte un commit în fiecare (referință: 3.1) prin intermediul comenzii `git branch` și denumirea branch-ului de care am avut nevoie și `git commit -am` și mesajul commitului. Este și posibilitatea de a crea și a te muta pe branchul create printr-o comandă `git checkout -b`. Cu ajutorul comenzii `git push origin` și denumirea la branch am încărcat schimbarile pe remote origin.

În al patrulea pas am resetat un branch, după ce am adăugat un fișier nou și i-am făcut commit, la commit-ul anterior (referință: 3.2) cu ajutorul comenzii `git reset HEAD~` [3].

În al cincilea pas am adăugat un fișier în .gitignore (referință: 3.3) ce a făcut comenzile git să nu îl considere ca un fișier din proiect, cu excepția unor comenzi ce forțează git cli să lucreze cu acest fișier.

În al șaselea pas am făcut merge la 2 branchuri (referință: 3.4) cu ajutorul comenzii `git merge` și denumirea la branch-ul cu care aș dori să fac merge [4].

În al șaselea pas am creat un conflict între 2 branchuri, schimbând același fișier apoi am făcut `git pull` unul din altul, ce a generat un conflict (referință: 3.5). Sunt mai multe metode de a fixa un conflict, unele din ele ar fi:

- Manual - selectarea manuală a informației care trebuie să rămână, și ștergerea celei inutile sau care nu mai este necesară
- Cu ajutorul comenzii `git checkout [-theirs—ours]` ce ar preoretiza schimbările în dependență de alegerea făcută [5].
- Cu ajutorul unor tool-uri care sunt prezente în unele IDE

Eu am ales varianta manuală (referință: 3.7) și o prefer mai mult ca a 2-a variantă pentru a înlătura riscurile de pierdere de date. Apoi am făcut commit cu fixarea făcută (referință: 3.8).

Pentru ultimul pas am creat un tag nou (referință: 3.9) cu ajutorul butonului `Draft a new release` de pe pagina <https://github.com/AScripnic/MIDPS-laboratories/releases>.

3.3 Imagini

```
MINGW64:/d/Universitate/MIDPS/MIDPS-laboratories

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (dev)
$ git branch test && git checkout test
Switched to branch 'test'

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (test)
$ git branch > current-branches.txt

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (test)
$ git add . && git commit -am "add current branches file"
[test 3dc5649] add current branches file
1 file changed, 3 insertions(+)
create mode 100644 current-branches.txt

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (test)
$ git checkout dev
Switched to branch 'dev'
Your branch is up-to-date with 'origin/dev'.

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (dev)
$ git branch test2 && git checkout test2
Switched to branch 'test2'

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (test2)
$ git status > current-status.txt

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (test2)
$ git add . && git commit -am "add current branch status file"
[test2 2a44404] add current branch status file
1 file changed, 7 insertions(+)
create mode 100644 current-status.txt

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (test2)
$ git checkout dev
Switched to branch 'dev'
Your branch is up-to-date with 'origin/dev'.

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (dev)
$ git push origin test && git push origin test2
Enter passphrase for key '/c/Users/Alex/.ssh/id_rsa':
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 313 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To git@github.com:AScripnic/MIDPS-laboratories.git
 * [new branch]    test -> test
Enter passphrase for key '/c/Users/Alex/.ssh/id_rsa':
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 429 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To git@github.com:AScripnic/MIDPS-laboratories.git
 * [new branch]    test2 -> test2

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (dev)
$
```

Figure 3.1 – Adaugarea a 2 branch-uri și câte un commit

```
Select MINGW64;d/Universitate/MIDPS/MIDPS-laboratories

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (revert)
$ git status
On branch revert
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   status.txt

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (revert)
$ git commit -am "add new file"
[revert 12da1c9] add new file
 1 file changed, 7 insertions(+)
 create mode 100644 status.txt

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (revert)
$ git status
On branch revert
nothing to commit, working directory clean

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (revert)
$ git reset HEAD~

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (revert)
$ git status
On branch revert
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    status.txt

nothing added to commit but untracked files present (use "git add" to track)

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (revert)
$
```

Figure 3.2 – Reset la commit anterior

```
MINGW64;d/Universitate/MIDPS/MIDPS-laboratories

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (ignore)
$ git status
On branch ignore
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    file-to-ignore.txt

nothing added to commit but untracked files present (use "git add" to track)

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (ignore)
$ printf '\nfile-to-ignore.txt' >> .gitignore

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (ignore)
$ git status
On branch ignore
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   .gitignore

no changes added to commit (use "git add" and/or "git commit -a")

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (ignore)
$
```

Figure 3.3 – Adăugarea unui fisier în .gitignore

```
MINGW64;d/Universitate/MIDPS/MIDPS-laboratories

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (dev)
$ git merge test
Merge made by the 'recursive' strategy.
 current-branches.txt | 3 +++
 1 file changed, 3 insertions(+)
 create mode 100644 current-branches.txt

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (dev)
$
```

Figure 3.4 – Merge la 2 branch-uri

```
MINGW64/d/Universitate/MIDPS/MIDPS-laboratories
Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (test)
$ git pull origin test2
Enter passphrase for key '/c/Users/Alex/.ssh/id_rsa':
From github.com:AScripnic/MIDPS-laboratories
* branch      test2      -> FETCH_HEAD
Auto-merging current-status.txt
CONFLICT (content): Merge conflict in current-status.txt
Automatic merge failed; fix conflicts and then commit the result.

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (test|MERGING)
$
```

Figure 3.5– Crearea unui conflict

```
On branch test2
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    current-status.txt

nothing added to commit but untracked files present (use "git add" to track)
<<<<<< HEAD
text
=====
othertext
>>>>>> 87b708894899773864a7f748fbb83d87b078b08f
```

Figure 3.6– Fișierul în care a avut loc conflictul

```
On branch test2
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    current-status.txt

nothing added to commit but untracked files present (use "git add" to track)
othertext
```

Figure 3.7– Fișierul fixat după conflict

```
MINGW64/d/Universitate/MIDPS/MIDPS-laboratories
Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (test|MERGING)
$ git commit -am 'fix conflict'
[test 09af979] fix conflict

Alex@DESKTOP-47D4LHO MINGW64 /d/Universitate/MIDPS/MIDPS-laboratories (test)
$
```

Figure 3.8– Commit după rezolvarea conflictelor

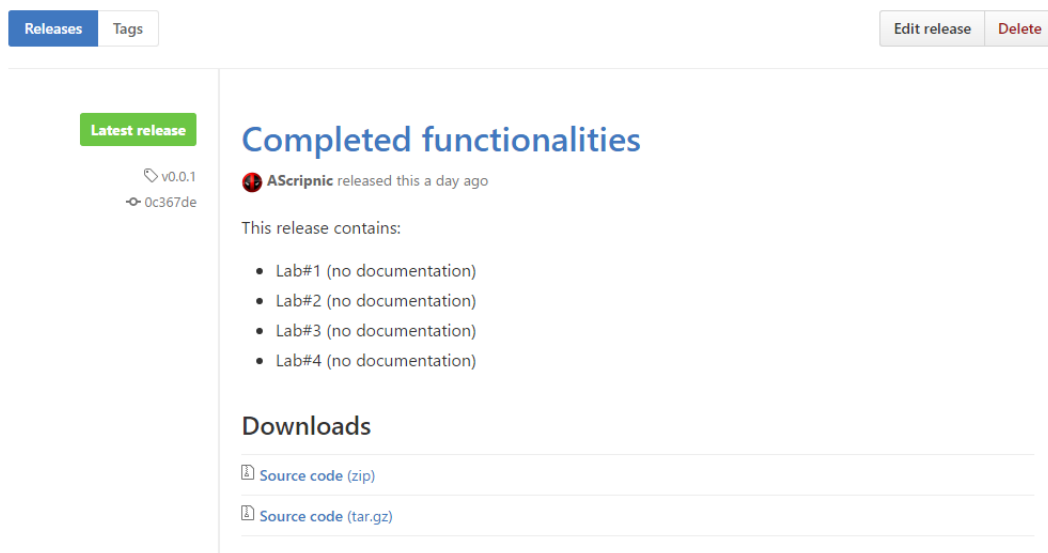


Figure 3.9– Crearea unui tag nou

Concluzie

Pe parcursul elaborării acestui laborator am studiat Version Control Systems, iar mai precis git și comenzile de bază în CLI.

Am descoperit noi metode de a face merge la un conflict între 2 branch-uri, cât și crearea, schimbarea, ștergerea unui branch. Am studiat posibilitatea de a face reset la un commit sau numai la un singur fișier.

References

- 1 Creating ssh key, *official page*, <https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/>
- 2 Set git cli config, *official page*, <https://help.github.com/articles/setting-your-username-in-git/>
- 3 reset command, <https://git-scm.com/docs/git-reset>
- 4 merge command, <https://git-scm.com/docs/git-merge>
- 5 fix conflict via checkout command, <http://gitready.com/advanced/2009/02/25/keep-either-file-in-merge-conflicts.html>